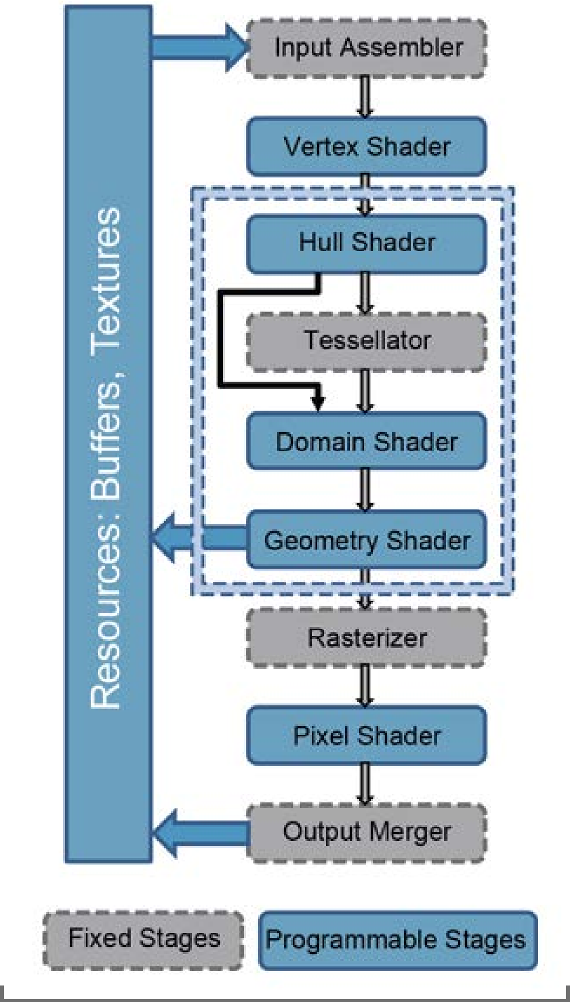
Scissor, Stencil, Depth Test

MEDIA DESIGN SCHOOL GAME DEV

Updated Graphics Pipeline

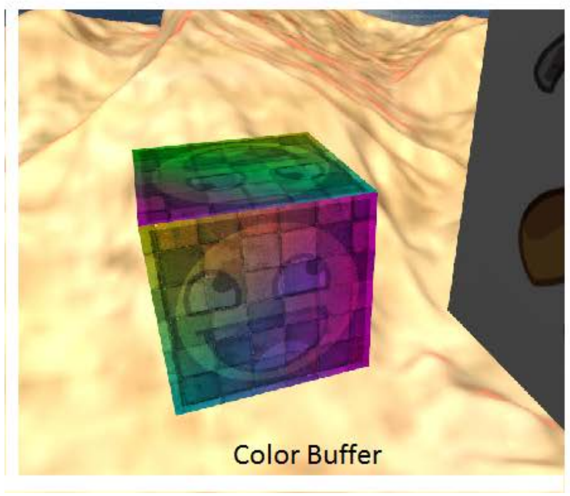


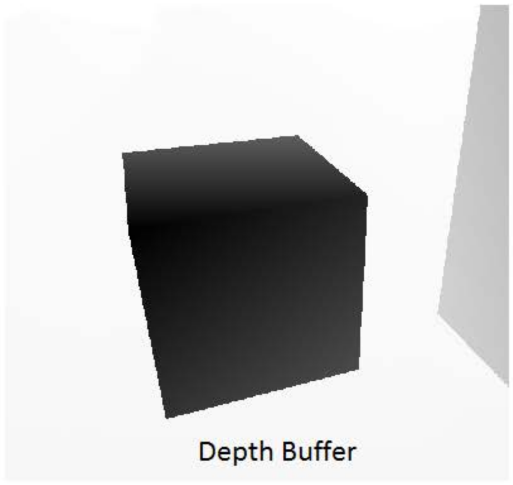
Output Merger Stage

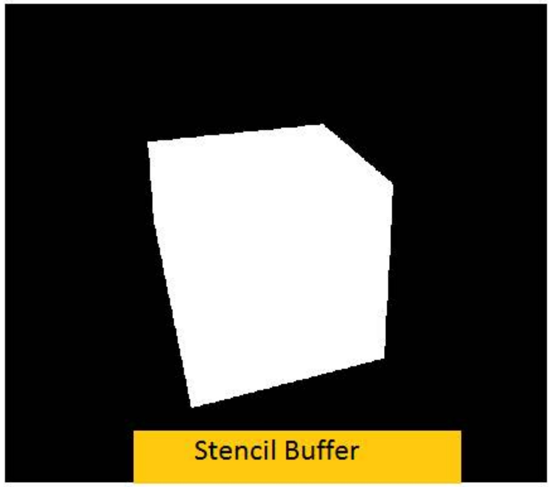
- In this stage, if needed, a post-shader Z/Stencil buffer test operation can be enabled to perform a final reject or depth replace.
- Pixels that are not rejected are written to the back buffer.
- Additionally, if multiple render targets (MRT) are used, the colors/alpha from all the targets are blended and written to th back buffer.

- Once the fragment shader has run, OpenGL needs to figure what do to with the fragments that are generate
- OpenGL then performs a number of other tests on the fragment to determine if and how it should be written to the framebuffer
- These tests (in logical order) are the
  - scissor test,
  - the stencil test,
  - and the depth test

Buffers







Scissor Test

- The scissor rectangle is an arbitrary rectangle that you can specify in screen coordinates which allows you to further clip rendering to a particular region
- Unlike in the viewport, geometry is not clipped directly against the scissor rectangle
- Rather individual fragments are tested against the rectangle as part of post-rasterization processing
- Some OpenGL implementations may apply scissoring either at the end of the geometry stage or in an early part of rasterization.

- If you wanted the scissor rectangle to be the same for all viewports, you could call glScissor()

```
void glScissor(GLint x,
               GLint y,
               GLsizei width,
               GLsizei height);
```

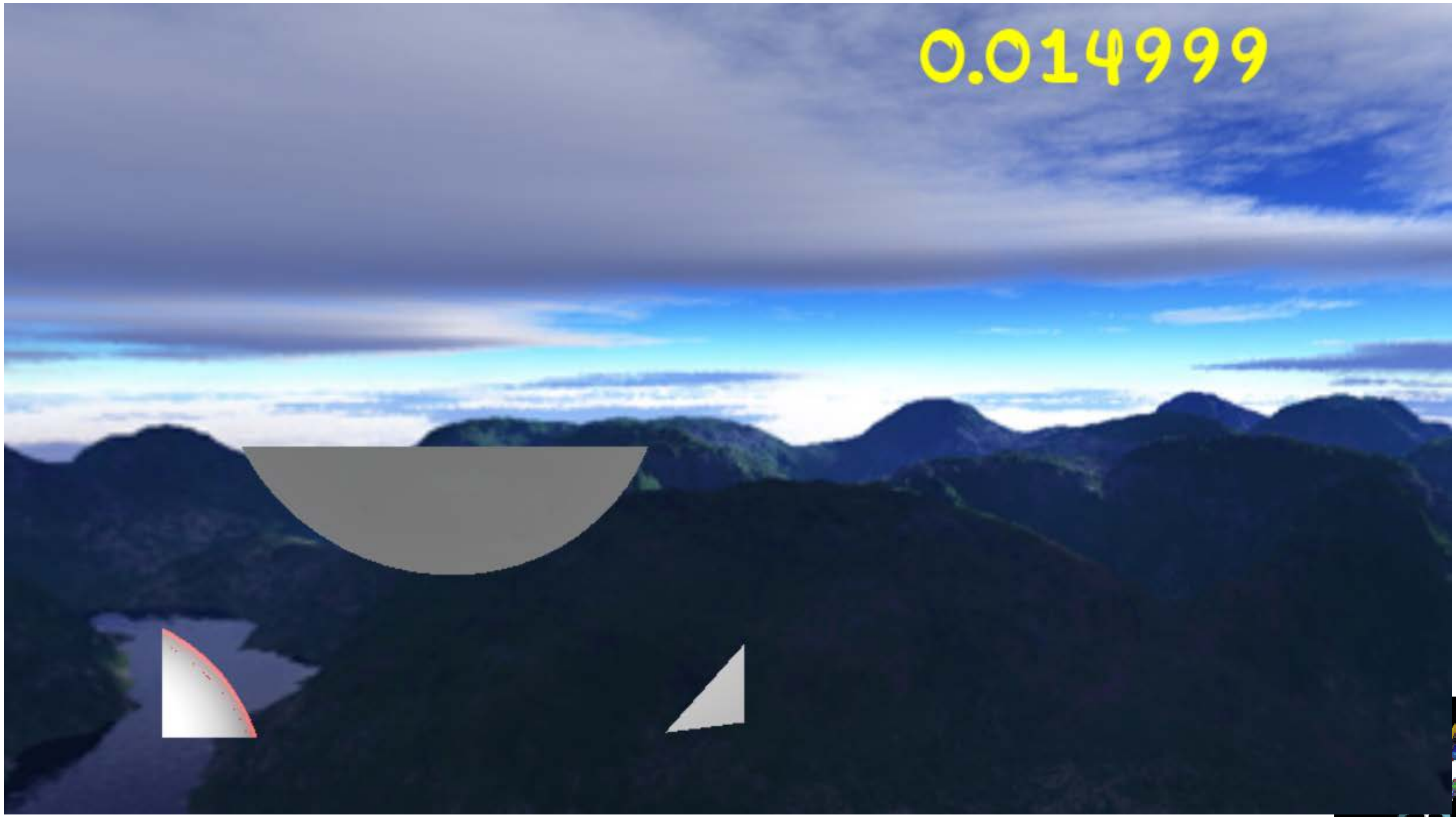
Scissor Test Example

- Enabling and disabling Stencil buffers
- Render function

```
// draw skybox

glEnable(GL_SCISSOR_TEST);
glScissor(200, 200, 400, 200);

//draw objects

glDisable(GL_SCISSOR_TEST);
```

MEDIA DESIGN SCHOOL GAME DEV

Scissor Test



MEDIA DESIGN SCHOOL GAME DEV

Depth Test

- After stencil operations are complete and if depth testing is enabled, OpenGL tests the depth value of a fragment against the existing content of the depth buffer
- If depth writes are also enabled and the fragment has passed the depth test, the depth buffer is updated with the depth value of the fragment.
- If the depth test fails, the fragment is discarded and does not pass to the following fragment operations.

Depth Test

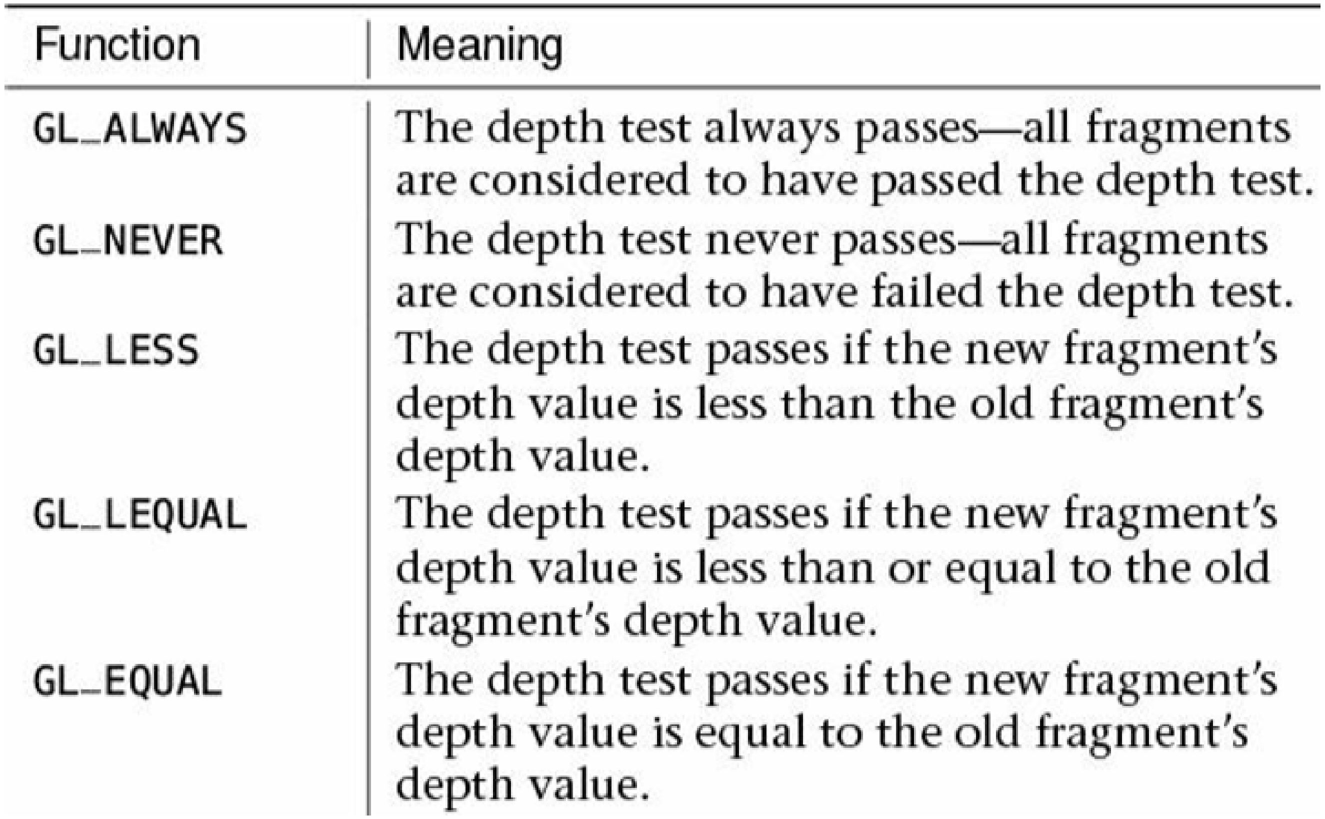
- The input to the primitive assembly stage is a set of vertex positions that make up primitives.
- Each has a z coordinate.
- This coordinate is scaled and biased such that the normal3 visible range of values lies between 0 and 1.
- This is the value that’s usually stored in the depth buffer.
- During depth testing, OpenGL reads the depth value of the fragment from the depth buffer at the current fragment’s coordinate and compares it to the generated depth value for the fragme being processed.

Depth Test

- You can choose which comparison operator is used to figure out if the fragment “passed” the depth test.
- To set the depth comparison operator (or depth function), call glDepthFunc().

```
void glDepthFunc(GLenum func);
```

Depth Test



Function	Meaning
GL_ALWAYS	The depth test always passes—all fragments are considered to have passed the depth test.
GL_NEVER	The depth test never passes—all fragments are considered to have failed the depth test.
GL_LESS	The depth test passes if the new fragment's depth value is less than the old fragment's depth value.
GL_LEQUAL	The depth test passes if the new fragment's depth value is less than or equal to the old fragment's depth value.
GL_EQUAL	The depth test passes if the new fragment's depth value is equal to the old fragment's depth value.

Depth Test

- The depth buffer is updated only when the depth test is enabled.

```
glEnable(GL_DEPTH_TEST);
```

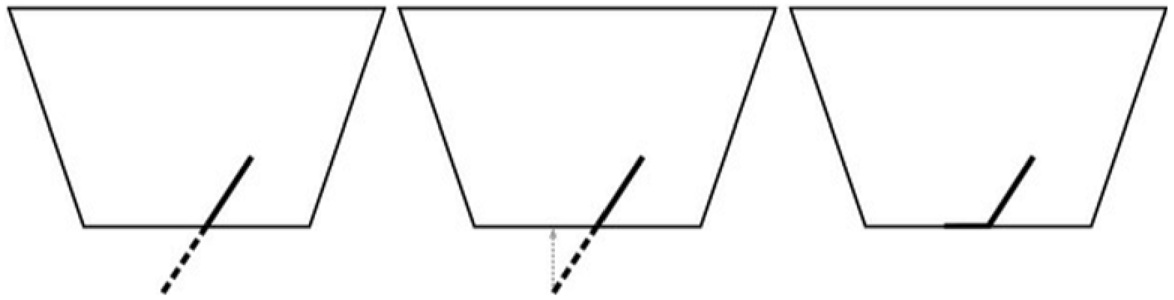
- To turn it off again, simply call glDisable() with the GL_DEPTH_TEST parameter.

Depth Clamping

- To eliminate the far plane and draw things at any arbitrary distance, we would need to store arbitrarily large numbers in the depth buffer—something that’s not really possible
- To get around this, OpenGL has the option to turn off clipping against the near and far planes and instead clamp the generated depth values to the range 0 to 1.
- glEnable(GL_DEPTH_CLAMP);
- glDisable(GL_DEPTH_CLAMP);

MEDIA
DESIGN
SCHOOL
GAME
DEV

Depth Clamping



Stencil Test

- The next step in the fragment pipeline is the stencil test
- You can think of the stencil test as cutting out a shape in cardboard and then using that cutout to spray-paint the shape on a canvas
- The spray paint hits the wall only in places where the cardboard is cut out
- If the pixel format of the framebuffer includes a stencil buffer, you can similarly mask your draws to the framebuffer

MEDIA DESIGN SCHOOL GAME DEV

Stencil Test

- Enabling and Disabling Stencil Test
  - glEnable(GL_STENCIL_TEST);
  - glDisable(GL_STENCIL_TEST);

- To control interactions with the stencil buffer, OpenGL provides two commands:
  - glStencilFunc()
  - glStencilOp()

MEDIA DESIGN SCHOOL GAME DEV

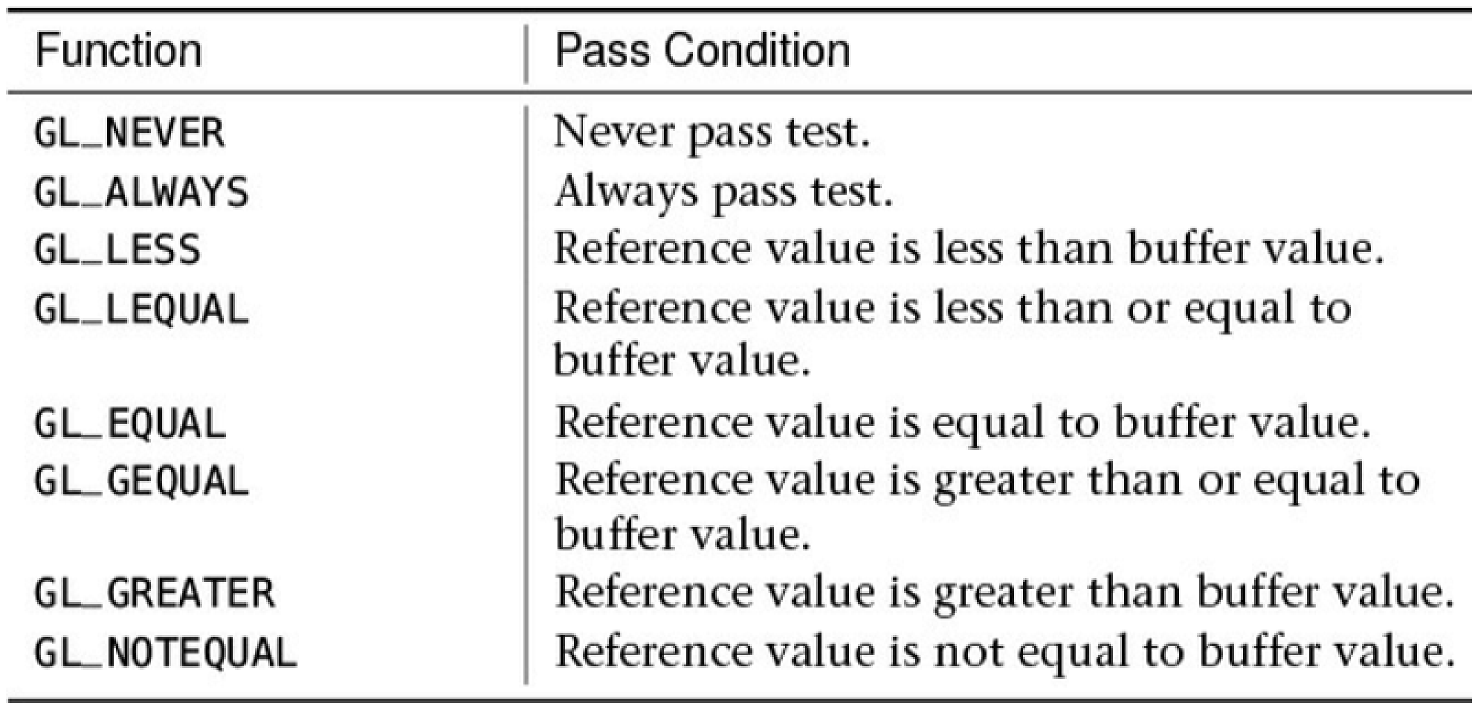
Stencil Test

```
void glStencilFunc( GLenum func,
                    GLint ref,
                    GLuint mask);
```

- *func* specifies under which conditions geometry will pass the stencil test.
- The *ref* value is the reference used to compute the pass or fail result
- the *mask* parameter lets you control which bits of the reference and the buffer are compared

MEDIA DESIGN SCHOOL GAME DEV

Stencil Test



Function	Pass Condition
GL_NEVER	Never pass test.
GL_ALWAYS	Always pass test.
GL_LESS	Reference value is less than buffer value.
GL_LEQUAL	Reference value is less than or equal to buffer value.
GL_EQUAL	Reference value is equal to buffer value.
GL_GEQUAL	Reference value is greater than or equal to buffer value.
GL_GREATER	Reference value is greater than buffer value.
GL_NOTEQUAL	Reference value is not equal to buffer value.

Stencil Test

- The next step is to tell OpenGL what to do when the stencil test passes or fails by using glStencilOp ()

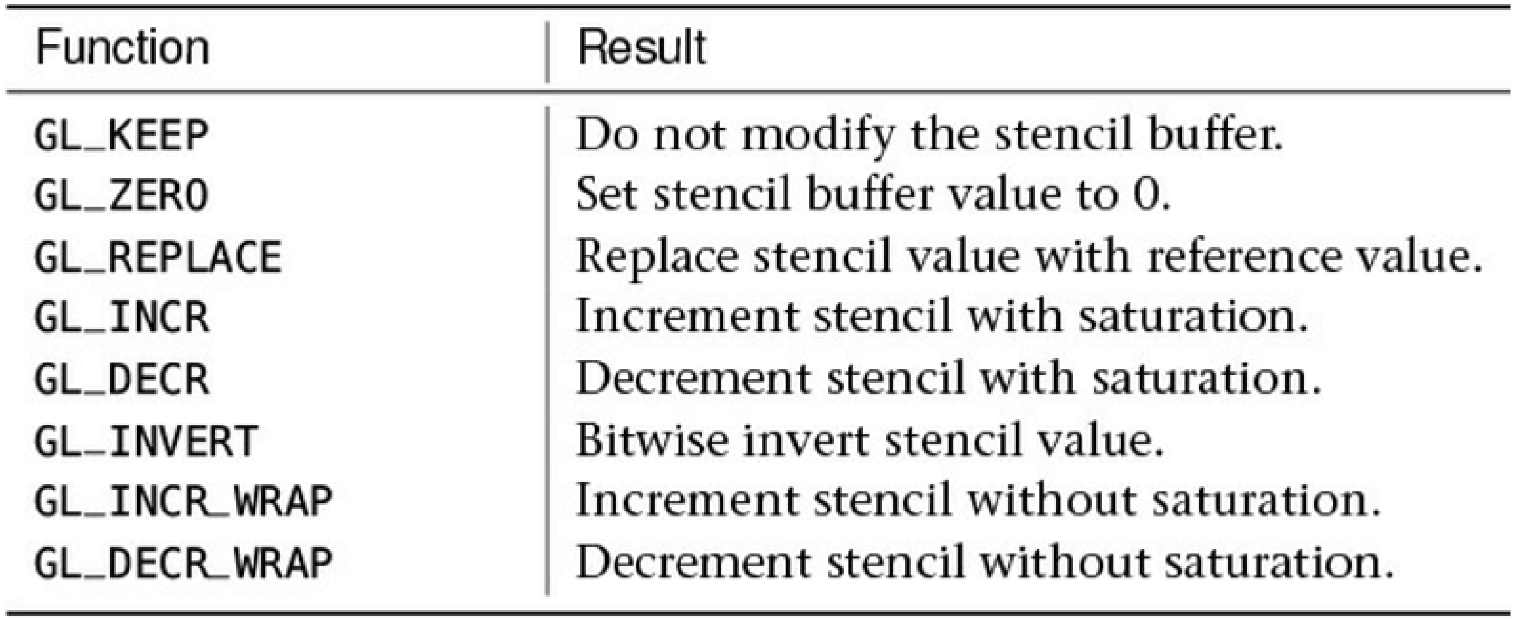
```
void glStencilOp(GLenum sfail,
                 GLenum dpfail,
                 GLenum dppass);
```

MEDIA DESIGN SCHOOL GAME DEV

Stencil Test

- *sfail*, is the action taken if the stencil test fails
- *dpfail* parameter specifies the action taken if the depth buffer test fails
- *dppass*, specifies what happens if the depth buffer test passes
- Because stencil testing comes before depth testing, should the stencil test fail, the fragment is killed right there and no further processing is performed

Stencil Test



Function	Result
GL_KEEP	Do not modify the stencil buffer.
GL_ZERO	Set stencil buffer value to 0.
GL_REPLACE	Replace stencil value with reference value.
GL_INCR	Increment stencil with saturation.
GL_DECR	Decrement stencil with saturation.
GL_INVERT	Bitwise invert stencil value.
GL_INCR_WRAP	Increment stencil without saturation.
GL_DECR_WRAP	Decrement stencil without saturation.

Stencil Test

- Stencil Mask
  - glStencilMask(0xFF); // Each bit is written to the stencil buffer as is
  - glStencilMask(0x00); // Each bit ends up as 0 in the stencil buffer (disabling writes)

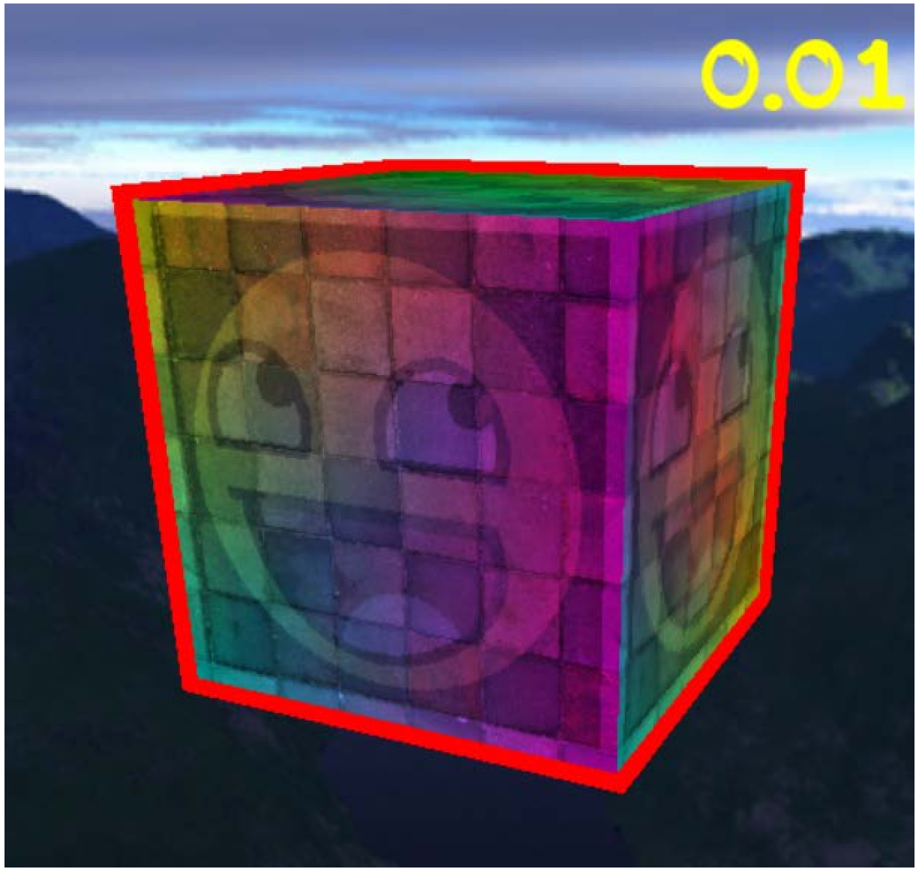
- Example
- Render function
- glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT);//clear stencil buff

Stencil Test

- glEnable(GL_STENCIL_TEST);//enable stencil buffer
- glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE); //stPass, dpFail, bothPass
- glStencilFunc(GL_ALWAYS, 1, 0xFF);//specify condition for stencil pass
- glStencilMask(0xFF);//enable writing to stencil buffer
- //Draw Object
- glStencilMask(0x00);//disable writing to stencil buffer
- glStencilFunc(GL_NOTEQUAL, 1, 0xFF);
- // Draw Scaled up Object
- glDisable(GL_STENCIL_TEST);

```
//enable stencil and set stencil operation
glEnable(GL_STENCIL_TEST);
glStencilOp(GL_KEEP, GL_KEEP, GL_REPLACE); //stPass, dpFail, bothPass

//** 1st pass **
//set current stencil value
glStencilFunc(GL_ALWAYS, // test function
              1,// current value to set
              0xFF);//mask value,

glStencilMask(0xFF);//enable writing to stencil buffer

//--> render regular sized cube // fills stencil buffer

// ** 2nd pass **
glStencilFunc(GL_NOTEQUAL, 1, 0xFF);
glStencilMask(0x00); //disable writing to stencil buffer

//--> render scaled up cube // write to areas where value is not equal to 1

//disable writing to stencil mask
glStencilMask(0x00);
glDisable(GL_STENCIL_TEST);
```

Stencil Test



Exercise

- Practice Scissor Test
- Practice Stencil Test
- What is Early Testing?
- What is Z – Fighting and how to prevent Z - Fighting